Code Baselining for the Land Information System
Submitted under Task Agreement GSFC-CT-2
Cooperative Agreement Notice (CAN)
CAN-00OES-01
Increasing Interoperability and Performance of Grand Challenge Applications in the Earth, Space, Life, and Microgravity Sciences

Version 1.0
Draft 1

July 26, 2002

Contents

List of Figures

List of Tables

1 Description of the Milestone

The milestone for the Global Land Data Assimilation System (GLDAS) code baseline deals with the implementation and execution of the Community Land Model (CLM) and the National Oceanic and Atmospheric Administration's NOAH (**N**ational Center for Environmental Prediction, **O**regon State University, United States **A**ir Force, and Office of **H**ydrology) land surface model (LSM) within the GLDAS at 1/4° resolution on the ESS Testbed for the near-term retrospective period. The milestone also requires publishing an initial version of documented source code made publicly available via the Web. The expected completion date is July 2002.

2 Description of the Test Case Problem

GLDAS is a software system that makes use of various satellite and ground based observation systems within a land data assimilation framework to produce optimal output fields of land surface states and fluxes. In addition to being forced with real time output from numerical prediction models and satellite and radar precipitation measurements, GLDAS derives model parameters from existing high resolution vegetation and soil coverages. The model results are aggregated to various time and spatial scales.

GLDAS includes ensembles of loosely coupled LSMs such as CLM and NOAH. These land surface models aim to characterize the transfer of mass, energy, and momentum between a vegetated surface and the atmosphere. The LSM predictions are greatly improved through the use of a data assimilation environment such as the one provided by GLDAS.

The Land Data Assimilation System (LDAS) system has been successful in modeling land surface processes in both real-time and at high resolution for North America. However, in order to accurately predict the land surface initialization and climate prediction problem, the LDAS system needs to be implemented globally at high resolution and at near-real-time. The increased resolution of the global modeling scale of the proposed Land Information System (LIS) significantly increases the computational requirements. As a result, this problem can be termed as a grand challenge problem.

The global land surface is modeled by dividing it into two-dimensional regions or cells (e.g. cells of size 1km x 1km, which would lead to approximately 50,000 times more grid points than that of GLDAS with cells of size $2^\circ \times 2.5^\circ$). Each cell can have a partial spatial coverage by a number of vegetation types, as well as bare soil. The vegetation characteristics such as leaf area index, stomatal resistance, etc. might be time varying. The conditions in each cell (energy, water fluxes, etc.) are computed at

different time intervals. Each cell is driven by different atmospheric forcing variables.

Assuming approximately 0.4 milliseconds for each LSM run on a particular cell, it can be estimated that modeling land surface processes over a year with 15 minute timesteps would require approximately 74 years of runtime. This problem is clearly a grand challenge simply from computational perspective.

The baselining results presented in this report were obtained by executing the GLDAS system on the following NASA AMES systems.

- HOPPER: SGI Origin 2000 IRIX64 6.5, 64 250MHz IP27 Processors
- LOMAX: SGI Origin 3000 IRIX64 6.5, 512 400MHz IP35 Processors

The domain resolution was set to be 1/4°. The CLM and NOAH LSMs were used in various runs. Two different timesteps (15 and 30 minutes) were used in all runs. For simplicity, only one tile per grid was considered in the runs. The output files were written using the GRIB format. Various combinations of forcings (including precipitation, shortwave, longwave radiation) were employed in the runs. The scalability of the code on a single processor at different domain resolutions was also examined.

3 Description of the Computer Code Used

This section provides an algorithmic description of the computer code used in the baselining. GLDAS is a model control and input/output system (consisting of a number of subroutines, modules written in Fortran 77 and 90 source code) that drives multiple offline one-dimensional LSMs using a vegetation defined "tile" or "patch" approach to simulate subgrid scale variability. The one-dimensional LSMs, which are subroutines of GLDAS, apply the governing equations of the physical processes of the soil-vegetation-snowpack medium. These equations are model independent.

3.1 Description of Algorithms

- **GLDAS:**
  Figure 1 shows the algorithmic steps involved in GLDAS. The user selects the model domain and spatial resolution, the duration and timestep of the run, the land surface model, the type of forcing from a list of model and observation based data sources, the number of "tiles' per grid square, the soil parameterization scheme, reading and writing of restart files, output specifications, and the functioning of several other enhancements including elevation correction and data assimilation.

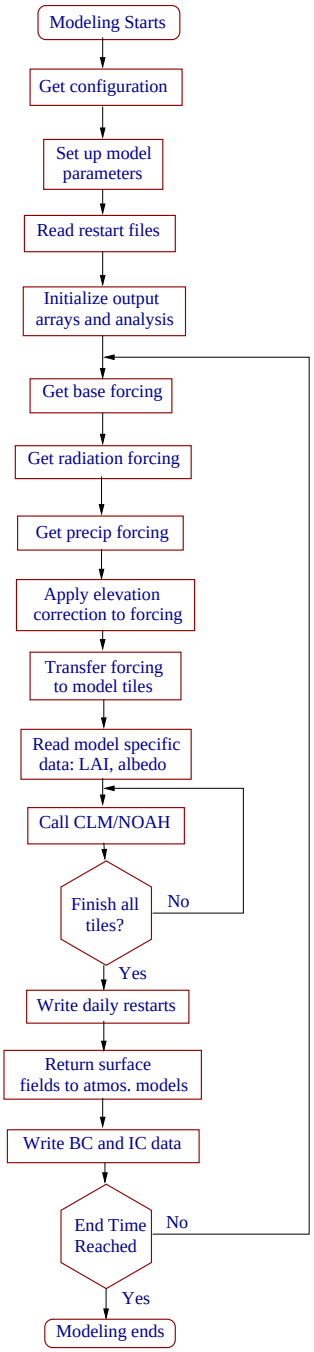

Figure 1: Flowchart for LDAS

Once the user specifications are read in, the system then reads the vegetation information and assigns subgrid tiles on which to run the one-dimensional simu-

lations. Memory is dynamically allocated to the global variables, many of which exist within Fortran 90 modules. The model parameters are read and computed next. The time loop begins and forcing data is read, time/space interpolation is computed and modified as necessary. The selected model is run for a vector of "tiles", intermediate information is stored in modular arrays, and output and restart files are written at the specified output interval.

- **CLM:**
  CLM is a 1-D land surface model with all forcings, parameters, dimensioning, output routines, and coupling performed by an external driver of the user's design (in this case done by GLDAS). The model applies finite-difference spatial discretization methods and a fully implicit time-integration scheme to numerically integrate the governing equations. The model subroutines apply the governing equations of the physical processes of the soil-vegetation- snowpack medium, including the surface energy balance equation, Richard's equation for soil hydraulics, the diffusion equation for soil heat transfer, the energy-mass balance equation for the snowpack, and the Collatz et al. formulation for the conductance of canopy transpiration.

- **NOAH:**
  NOAH land surface model is a stand-alone, uncoupled, 1-D column model. In this uncoupled mode, near-surface atmospheric forcing data is required as input forcing. The LSM simulates soil moisture (both liquid and frozen), soil temperature, skin temperature, snowpack depth, snowpack water equivalent, canopy water content, and the energy flux and water flux terms of the surface energy balance and surface water balance. The model applies finite-difference spatial discretization methods and a Crank-Nicholson time-integration scheme to numerically integrate the governing equations of the physical processes of the soil vegetation-snowpack medium, including the surface energy balance equation, Richard's equation for soil hydraulics, the diffusion equation for soil heat transfer, the energy-mass balance equation for the snowpack, and the Jarvis equation for the conductance of canopy transpiration.

3.2 Documentation of the Computer Code

The documentation of GLDAS and the land surface models (CLM 1.0 and NOAH 2.5) can be accessed at `http://lis.gsfc.nasa.gov/docs/LDAS-Doc/ldas2/index.html`.

3.3 Code Repository

The computer source code employed in the baselining may obtained from the LIS baseline repository .

4 Results

GLDAS was run on different SGI Origin systems with various combinations of forcings and different land surface models. The computational demands of various runs are quantified using three parameters: CPU times, disk usage, and memory usage.

4.1 CPU Times

A dynamic runtime profiling, using SGI's speedshop toolkit, was conducted to identify the most computationally intensive features of the code. Figures 2, 3, 6, and 7 show the CPU times of these functions. They are identified as:

- `getgeos`: This function opens, reads, and interpolates GEOS (Goddard Earth Observing System) forcing.
- `ipolates`: This function performs spatial interpolation.
- `zterp`: This function computes the zenith angle based temporal interpolation.
- `getgrad`: This function opens, reads, interpolates, and overlays radiation forcing.
- `getglbpcp`: This function opens and reads global precipitation forcing.
- `clm_main`: This is the main call to the CLM LSM.
- `clm_out`: CLM output writer.
- `noah_main`: This is the main call to the NOAH LSM.
- `noah_out`: NOAH output writer.

The corresponding contributions to the CPU times are shown as percentages in Figures 4, 5, 8 and 9. GEOS, NRL, and AGRMET indicate GEOS forcing, NRL precipitation forcing, and AGRMET radiation forcings, respectively. TS=1800 denote a timestep of 30 minutes.

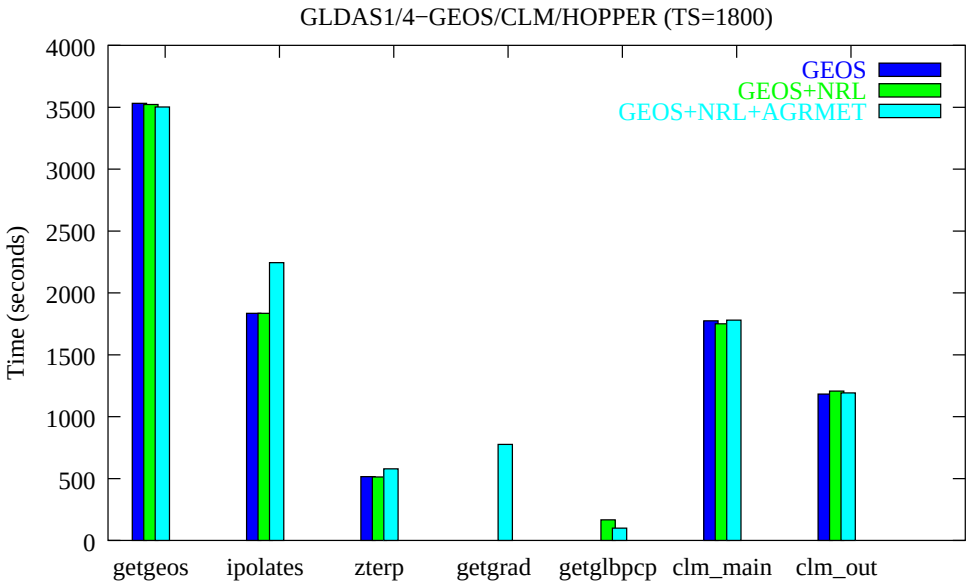

Figure 2: CPU times of computationally intensive functions on HOPPER for CLM runs

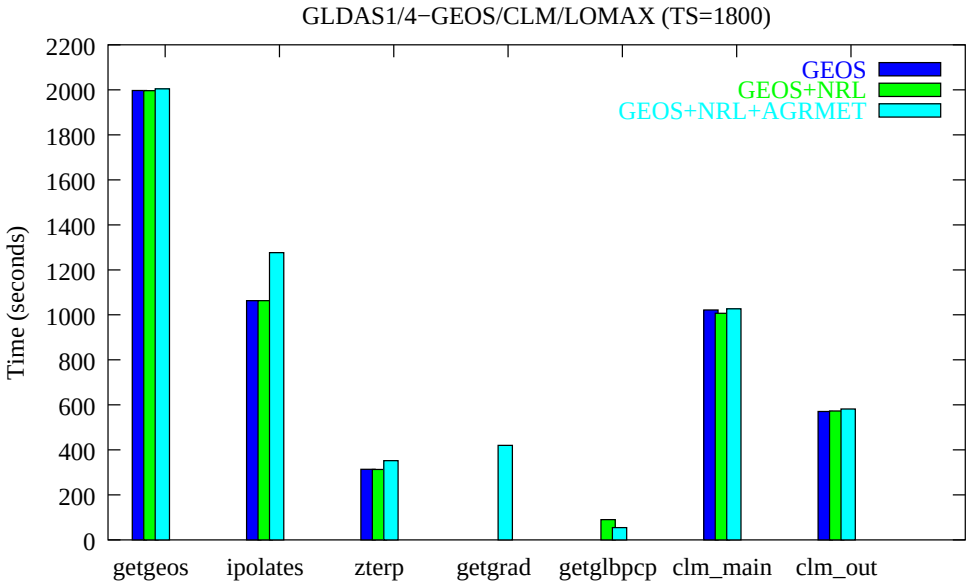

Figure 3: CPU times of computationally intensive functions on LOMAX for CLM runs

It can be observed that the contributions of functions in terms of percentages remain consistent across different computers, although the actual computational times

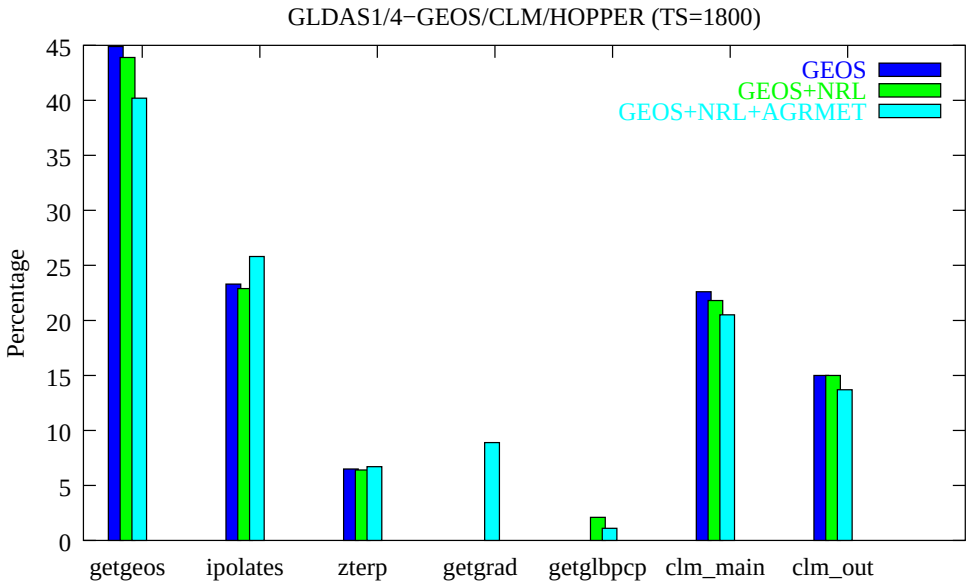

Figure 4: Percentage of total CPU times for computationally intensive functions on HOPPER for CLM runs

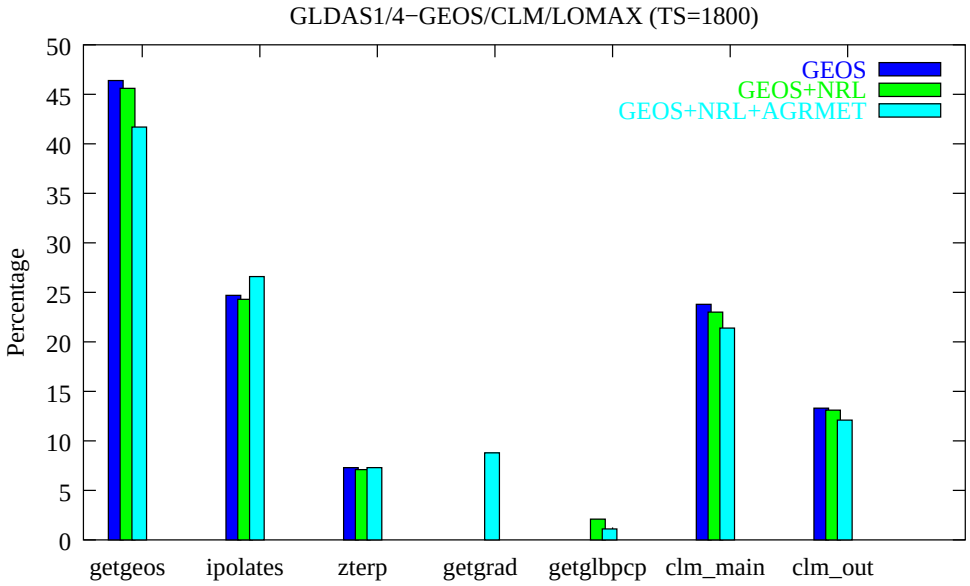

Figure 5: Percentage of total CPU times for computationally intensive functions on LOMAX for CLM runs

differ.

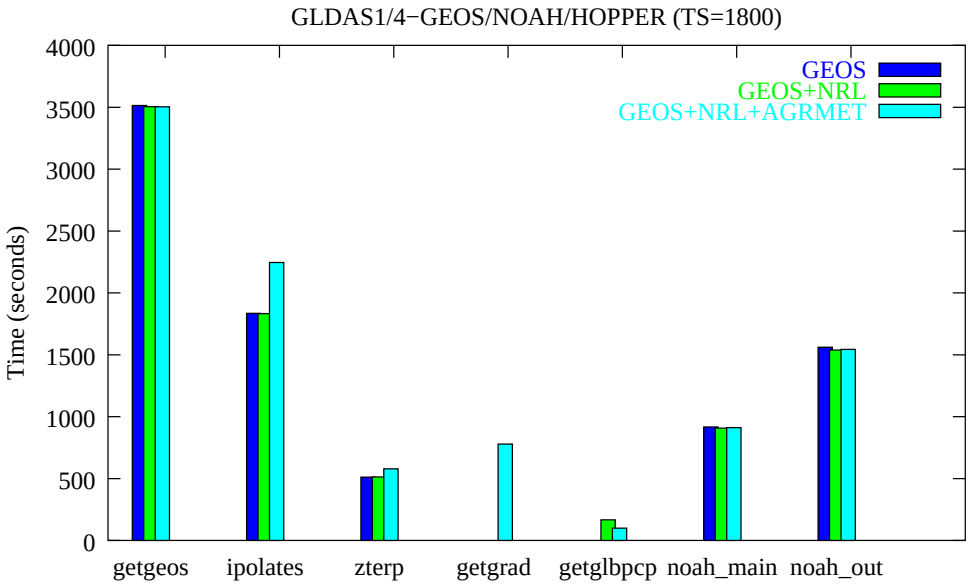

Figure 6: CPU times of computationally intensive functions on HOPPER for NOAH runs

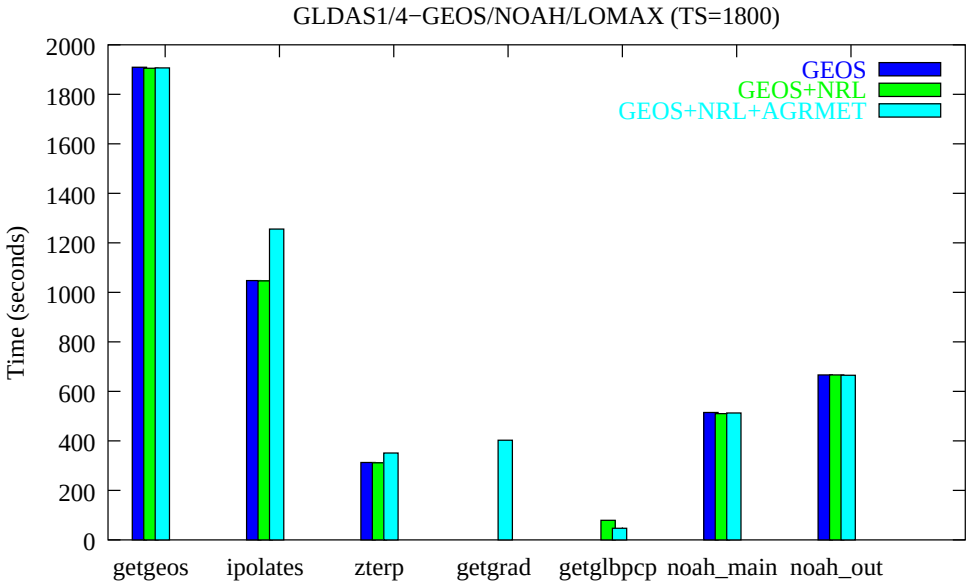

Figure 7: CPU times of computationally intensive functions on LOMAX for NOAH runs

The impact of the duration of timestep was examined by reducing the timestep from 30 minutes to 15. The results obtained using CLM on HOPPER is shown as

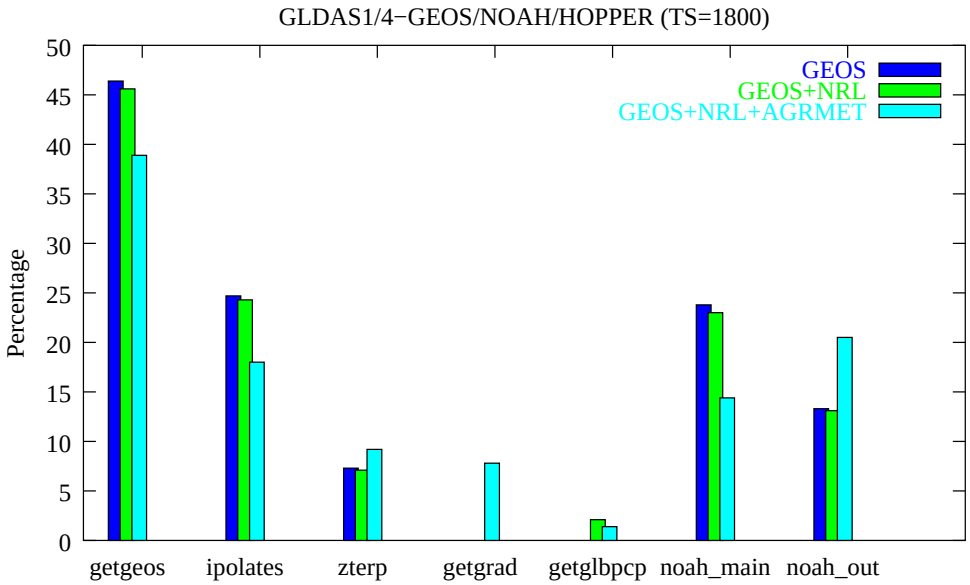

Figure 8: Percentage of total CPU times for computationally intensive functions on HOPPER for NOAH runs

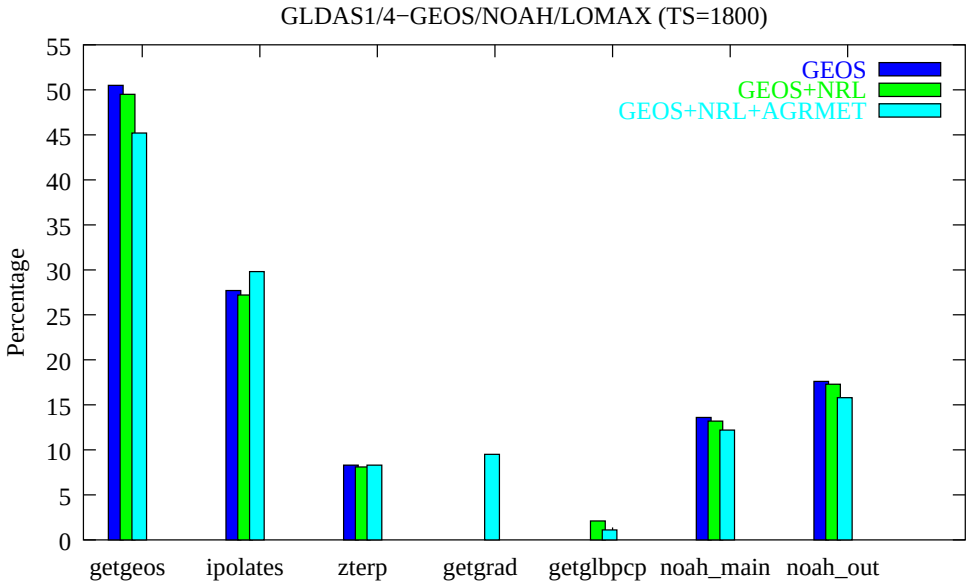

Figure 9: Percentage of total CPU times for computationally intensive functions on LOMAX for NOAH runs

a representative sample in Figure 10. The values obtained at 15 minute timesteps are compared with projected values using the computational times obtained using

30 minute timesteps. It can be observed that the time for the LSM calls increase (2 fold) with reduction in timestep duration. The interpolation functions (`ipolates` and `zterp`) are not affected by the decreased timestep since the data is not interpolated at every timestep. This leads to the less than linear scaling of the main forcing function (`getgeos`).

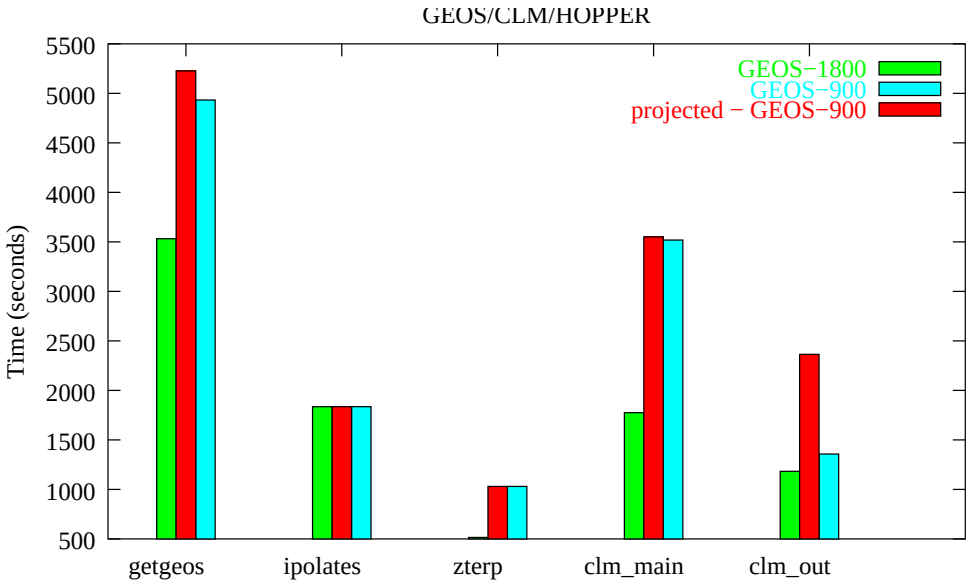

Figure 10: Effect of timestep duration on the computational times for CLM runs

To study the scalability of the code with increase in domain size, profiling studies were conducted for a domain increase from $2^\circ \times 2.5^\circ$ to $1/4^\circ$. The results are shown in Figures 11 and 12 for runs on HOPPER. The computational times at $1/4^\circ$ are also compared with some projected values computed from the $2^\circ \times 2.5^\circ$ runs. It can be observed that the different segments of the code scales mostly as expected with the output routines of the LSMs being the notable exceptions. This could be attributed to the nonlinear scaling nature of the specific output libraries in these routines.

Table 1 lists a measure of computational intensity on different platforms for various runs. It can be observed that the performance of the code on LOMAX is significantly better than that on HOPPER. For example, the computational complexity measured for the CLM run using GEOS forcing on HOPPER is approximately twice that of the run on LOMAX. It can also be seen that CLM is computationally more intensive than NOAH. The measured computational complexities for NOAH are smaller than that of CLM as shown in Table 1. From the Figures 2, 3, 6, and 7 , it can also be observed that the calls to CLM routines take more time than those of NOAH.

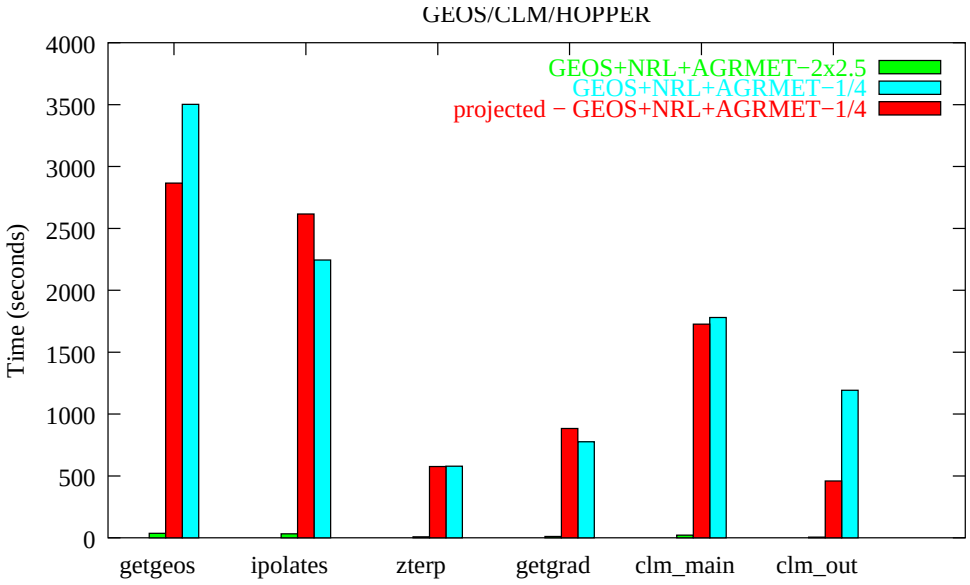

Figure 11: Effect of domain increase from $2° \times 2.5°$ to $1/4°$ on the computational times for CLM runs

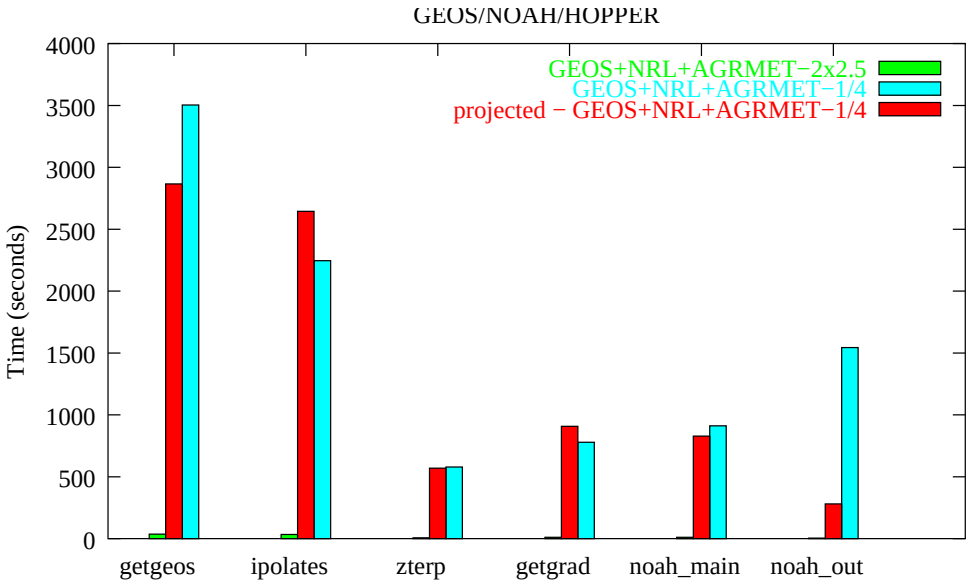

Figure 12: Effect of domain increase from $2° \times 2.5°$ to $1/4°$ on the computational times for NOAH runs

Table 1: Measure of computational intensity for GLDAS 1/4° runs (ms/gridcell/day)

		CLM		NOAH	
		Timestep (minutes)			
		15	30	15	30
HOPPER	GEOS	13.9	9.1	13.7	8.3
	GEOS + NRL	14.2	9.3	14.1	8.5
	GEOS + NRL + AGRMET	15.2	10.1	14.6	9.3
LOMAX	GEOS	7.7	5.0	6.9	4.4
	GEOS + NRL	7.9	5.1	7.1	4.7
	GEOS + NRL + AGRMET	8.4	5.6	7.6	4.9

4.2 Disk Usage

The GLDAS code uses three categories of global data: parameter data, input forcing data and output data. The parameter data include vegetation classification, land mask, etc., with a size of approximately 136GB. The code reads in the forcing data at regular intervals, with the traffic estimated to be approximately 279 MB/day.

For the baselining results presented in this report, the code and requisite files require 1.1GB of hard disk space. The disk space required for output for different baselining runs are shown in Table 2. It can be noticed that the disk usage increases almost linearly with the increase in domain size. Using these values, the output data volume for the global 1km run using CLM and NOAH can be estimated to be approximately 250 GB/day and 150 GB/day, respectively.

Table 2: Disk Usage for various GLDAS runs (in MB)

	CLM	NOAH
GLDAS 2° × 2.5°	5	3
GLDAS 1/4°	400	235

4.3 Memory Usage

The GLDAS code also requires significant memory for execution. The following table 3 lists the approximate memory requirements for GLDAS runs with different land

Table 3: Memory Usage for various GLDAS runs

	CLM	NOAH
GLDAS $2^\circ \times 2.5^\circ$	250 MB	200 MB
GLDAS $1/4^\circ$	3.5 GB	2.0 GB

surface models. Using these values, the memory requirements for GLDAS run at 1km resolution on a single machine can be estimated to be approximately 2TB.

5 Conclusions

These profiling results have demonstrated the functions that are most time-consuming, thereby identifying the portions of our code-set that require our immediate attention. These profiling results also demonstrate that these critical functions scale with respect to time and space as predicted, suggesting that across-the-board performance improvements can be made from re-writing these critical routines, instead of having to make specialized performance improvements for specific scenarios. As discussed in the eariler section, Figure 10 shows the scalability of various code segments with timestep and Figures 11 and 12 shows the scalability of the code for a larger domain.

The baselining study has also helped in quantifying the disk and memory usage requirements of the GLDAS code. As mentioned earlier, the estimated disk output volume at 1km resolution (150-250GB/day) is extremely large and it is not feasible to store the output in a single file. As a result, the computing strategy must involve a design to distribute the data across different nodes to keep the output data volume manageable. Similarly, the projected memory usage at 1km resolution is very large ($\sim$ 2TB) and the problem need to be split up into smaller pieces to satisfy the memory requirements of a real-time operation.

6 Future Directions

In order to meet future milestones F and G, regarding the performance of LIS, we need to be able to characterize the behavior of our initial code set and use this characterization to guide our software development, system design, and code improvement. From the results presented in this report, it is apparent that global scale land surface modeling at 1km resolution poses significant computational challenges, from a computational as well as data/memory management perspectives. Parallel comput-

ing has emerged as the enabling technology that will help modern computers satisfy increasing high performance computing requirements. We plan to build a system that takes advantage of scalable parallel computing technologies to facilitate global land surface modeling. The land surface processes have rather weak horizontal coupling on short time and large space scales, enabling highly efficient scaling across massively parallel computing platforms. The high data densities could pose limitations on the land surface modeling efficiencies, and the LIS system will explore the use of high performance technologies to eliminate this bottleneck.